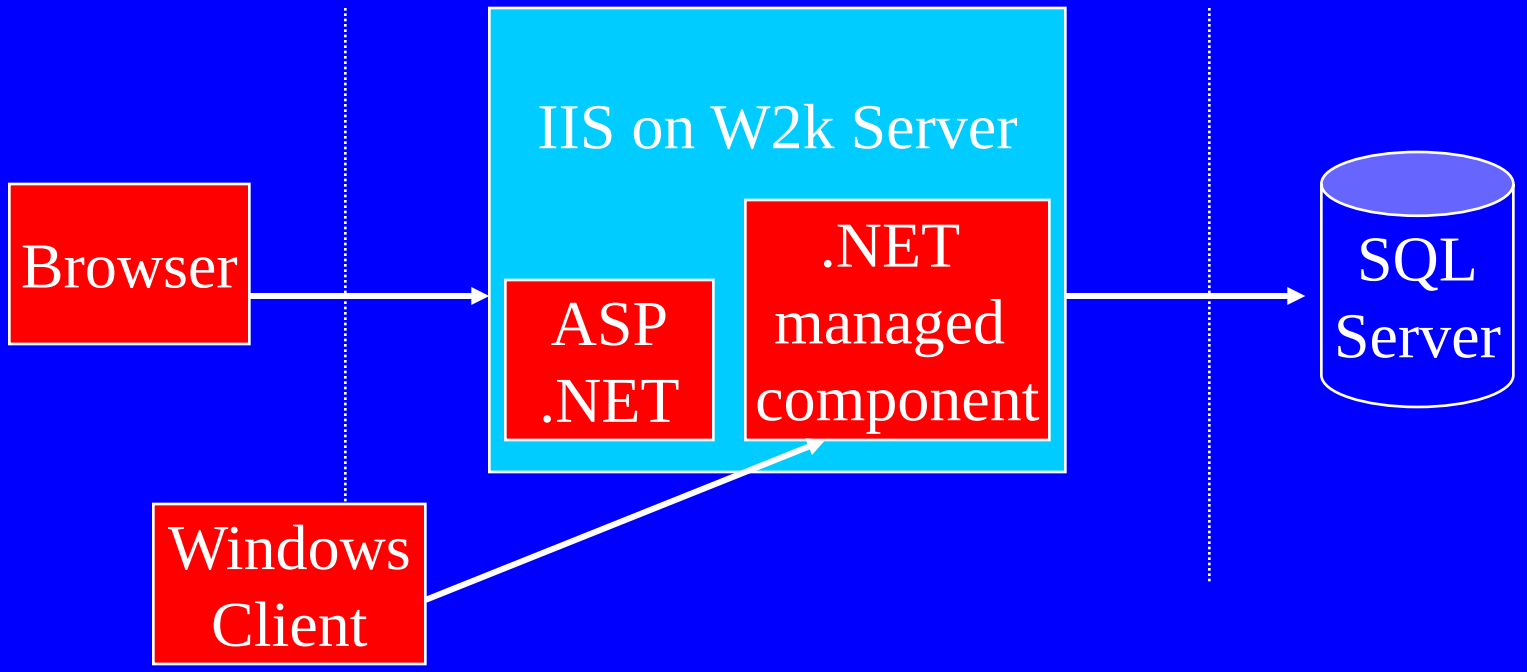
For Mr Olivier CARTON

Your student / Abdelrahman ELGAMAL

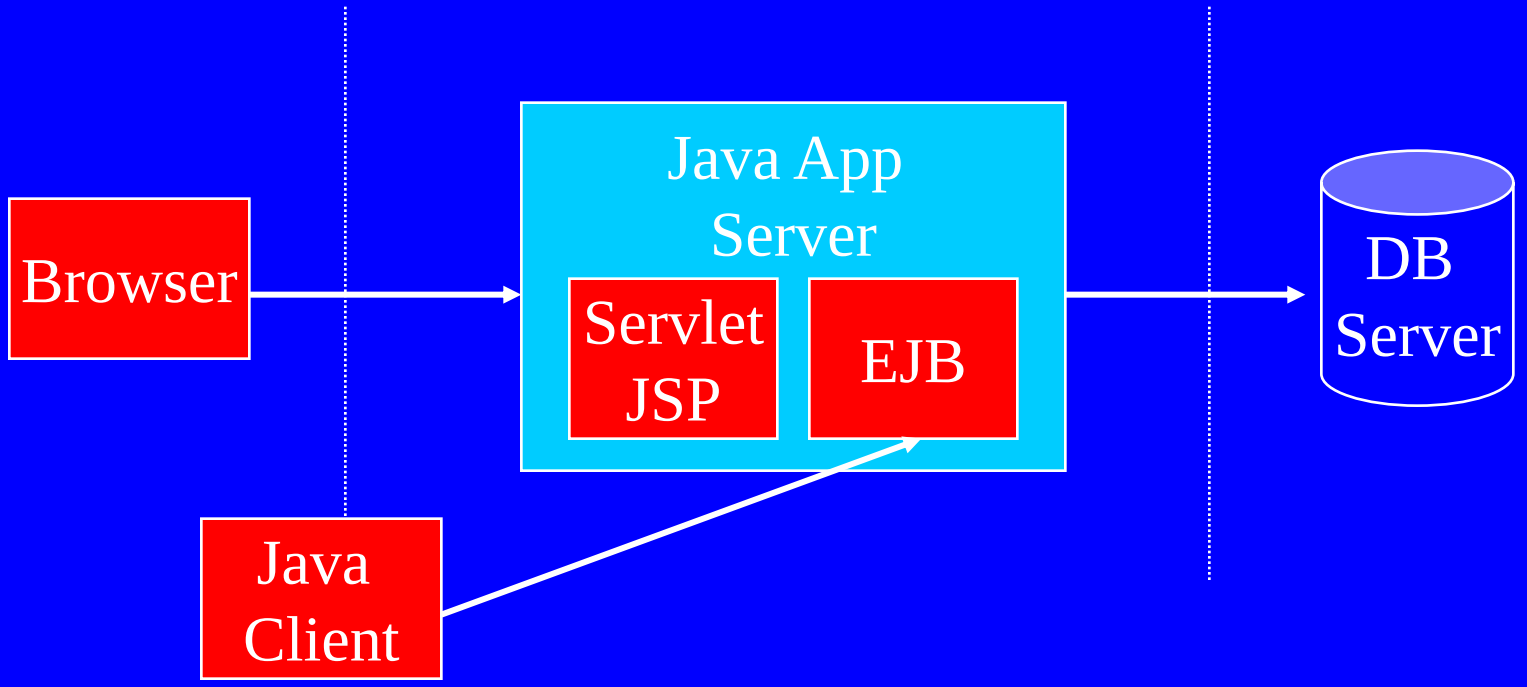
Java vs. .Net

Bent Thomsen
bt@cs.auc.dk
Department of Computer Science
Aalborg University

A typical .NET Enterprise Solution



A typical J2EE Enterprise Solution



Java vs .Net Solutions

- Both multi-tiered, similar computing technologies
- Both support “standards”
- Both offer different tools & ways to achieve the same goal.
- A lot of parallelism can be seen.
- Very difficult to compare and qualify the comparison because each has its own advantages & disadvantages.

The TMC Petshop Performance Case Study

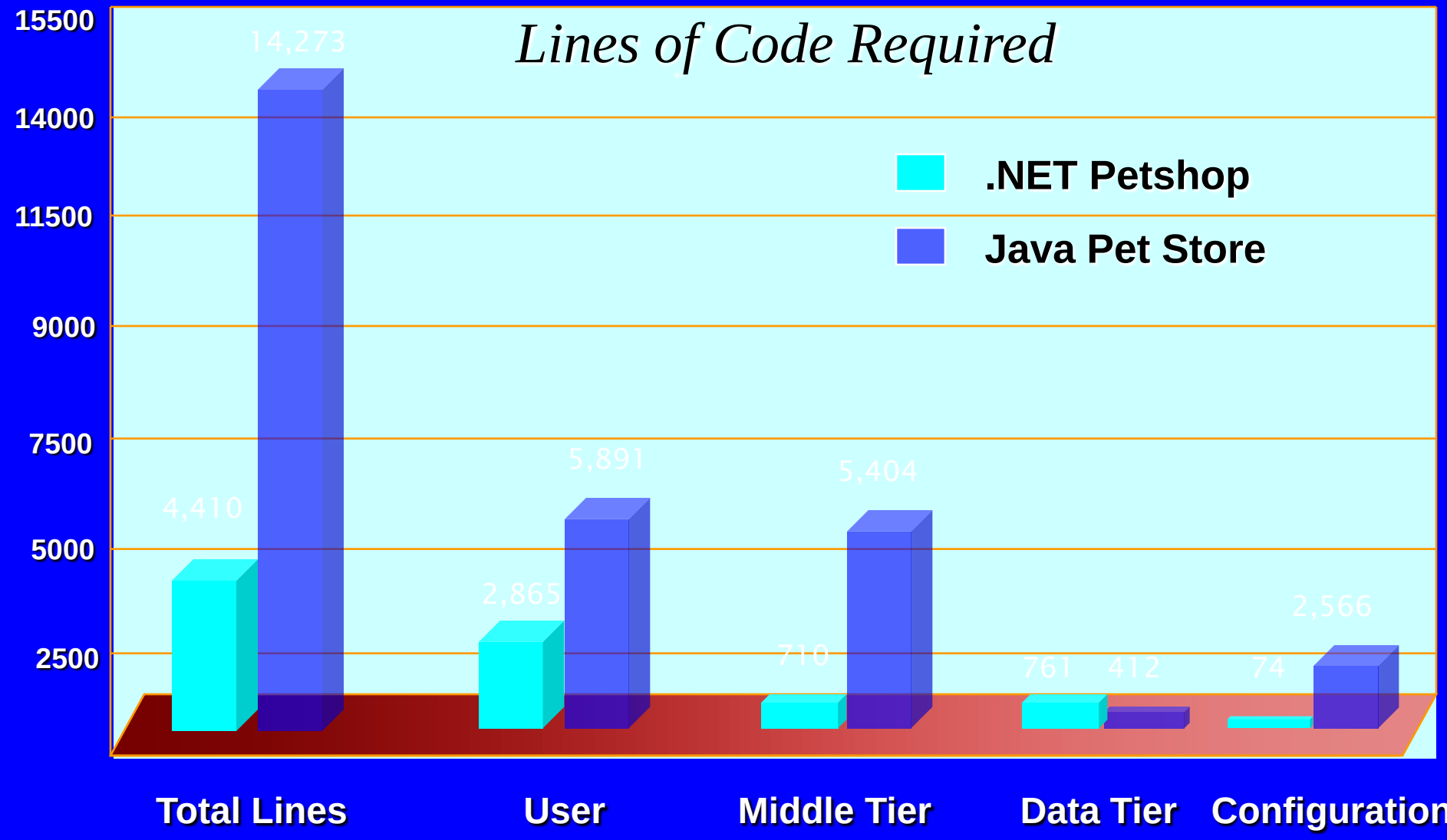
- Java Pet Store is Sun's primary blueprint application for J2EE
  - Source: http://java.sun.com/j2ee/blueprints
  - Illustrates best coding practices for J2EE
  - Ships as a sample application in IBM Websphere, Oracle Application Server 9i, Sun iPlanet, and BEA WebLogic
- The .NET Petshop is a port of the J2EE Java Pet Store to .NET
  - Source: http://www.gotdotnet.com/compare
  - Implements the same functionality as Java Pet Store
  - Illustrates best coding practices for .NET Framework
- In the TMC Petshop Performance Case Study, The Middleware Company implemented both the Java Pet Store and the .Net Petshop.
  - The J2EE version ran on two different application servers
  - All versions used the same hardware and OS

Java Pet Store Components

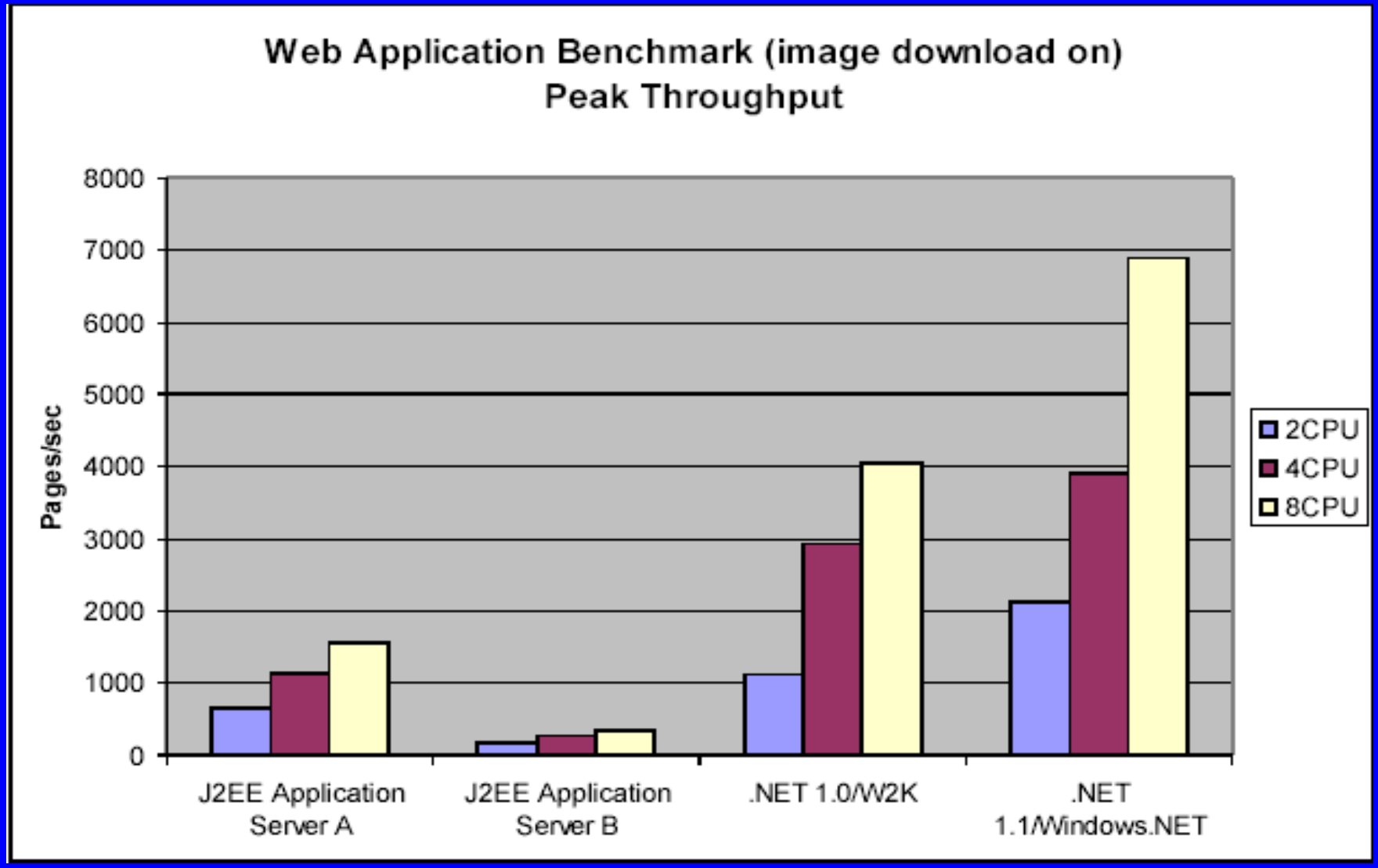
- The Storefront presents the main user interface in a Web front-end. Customers use the Storefront to place orders for pets.
- The Order Processing Center (OPC) receives orders from the Storefront.
- The Supplier fulfills orders from the OPC from inventory and invoices the OPC.
- The Admin presents the administrator interface in a JFC/Swing front-end. Administrators use the Admin to examine pending orders and approve or deny them.

Java Pet Store vs. .Net Pet Shop

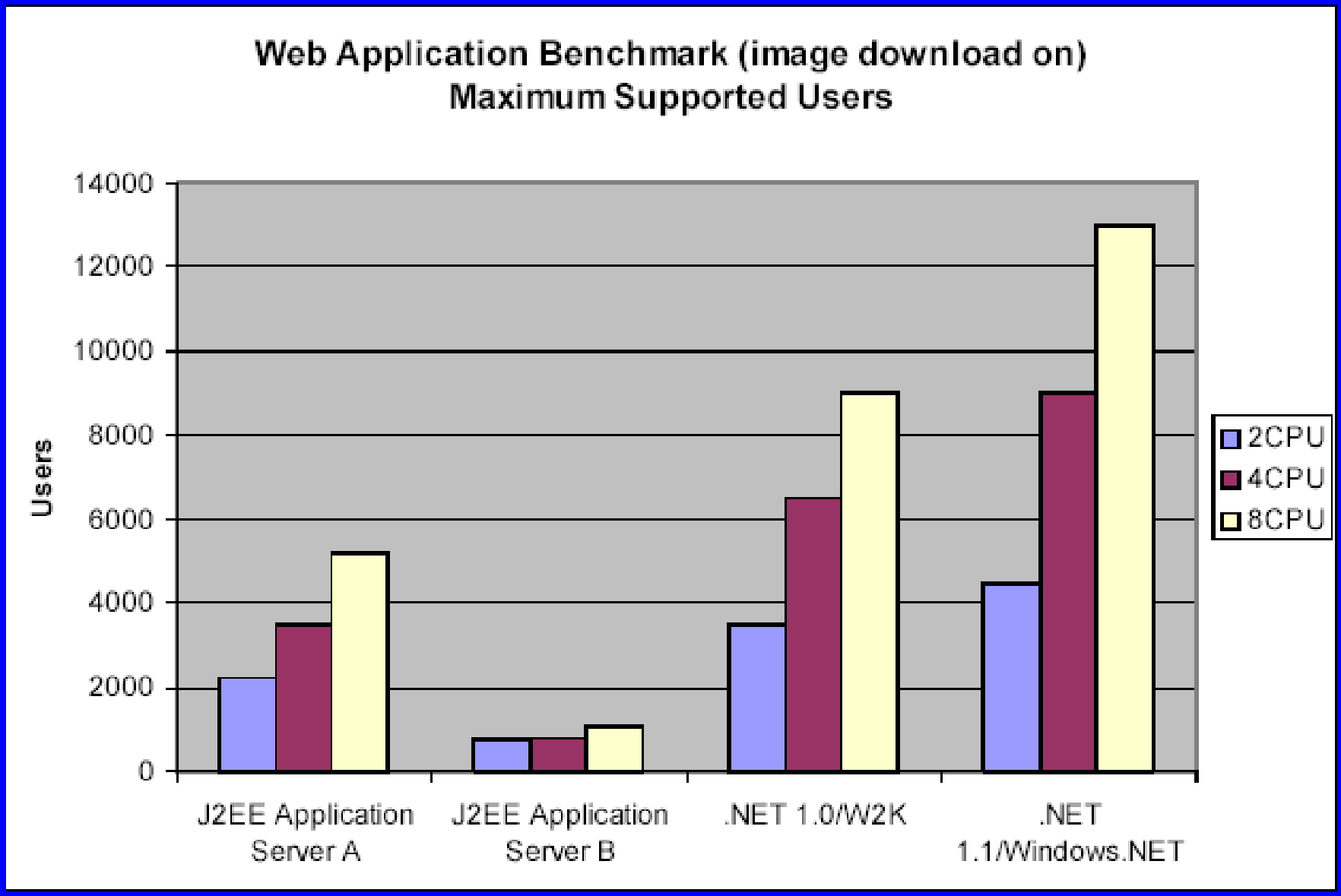
Porting Java Pet Store to .NET



TMC Pages per Second

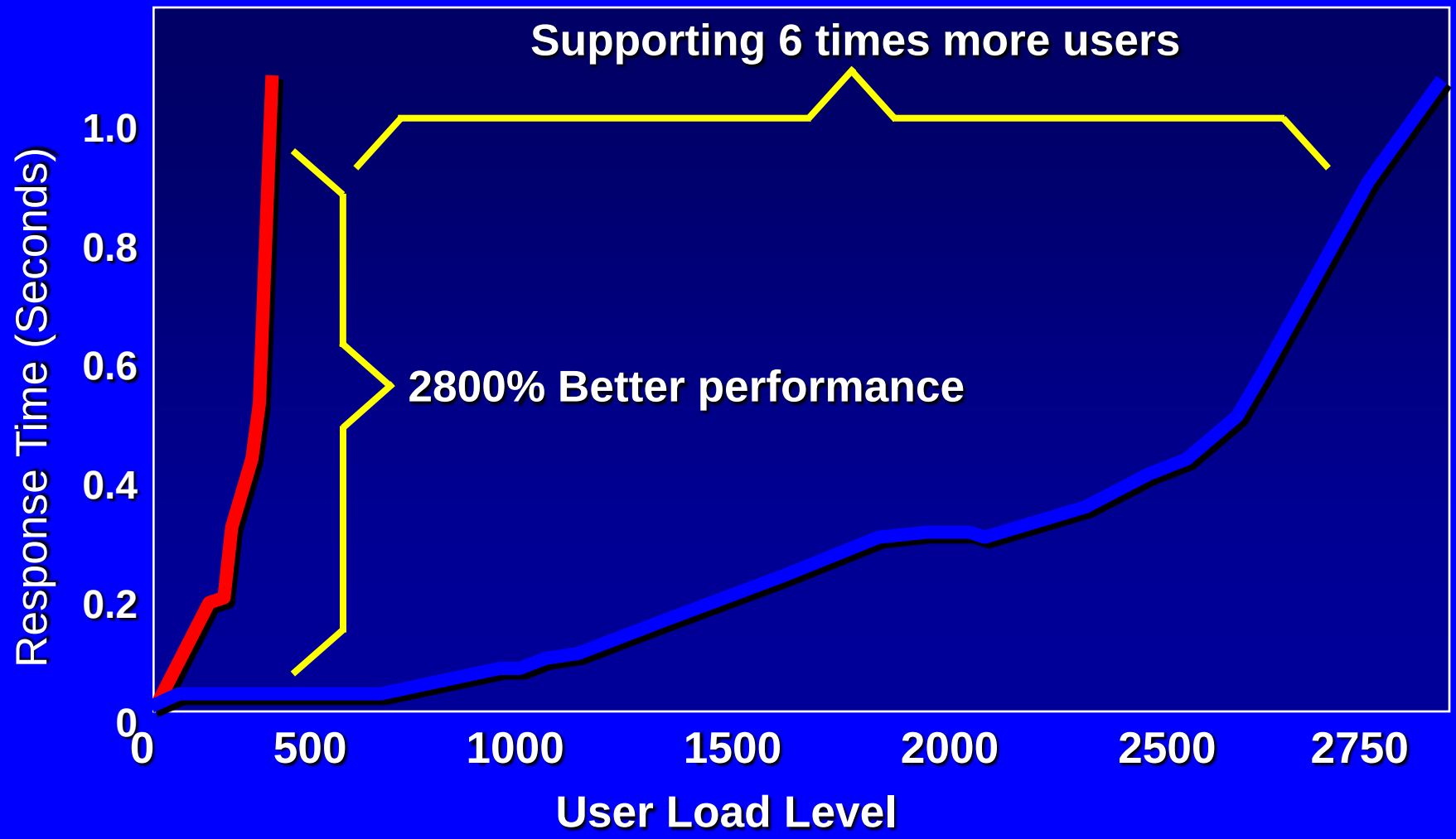


TMC Max Supported Users



How Microsoft interprets the data

- Based on Oracle-published data for tuned version of Java Pet Store
- Using Oracle's test scripts from their "9i App Server Challenge"
- Run on equivalent hardware



The “Conclusions”

- Microsoft concludes: ".NET is 28 times faster than J2EE"
  - Because:
  - “Java PetShop” is Sun’s “best practice” example
  - Numbers compared with published Oracle results
  - Identical functionality, all code published and documented
- Can we believe the raw numbers? Yes!
  - Why? Microsoft delivers full docs on the entire scenario

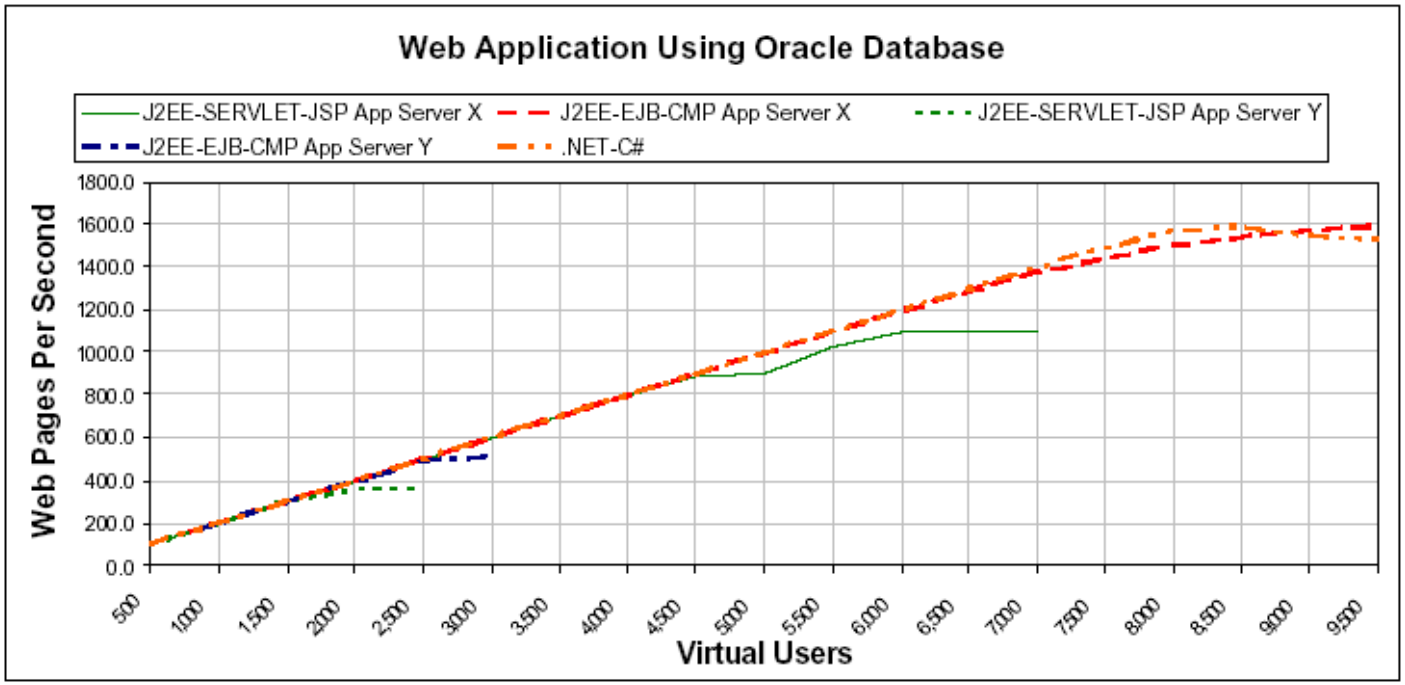
- But 28 times really?

The Shootout afterwards

- Java PetShop: J2EE blueprint application
  - Built by Sun to show "best practice" design
  - Implementation by TMC missed quite a few optimizations
- .NET PetShop
  - Built by TMC (with help from MS) with *different* design
    - Plain classes instead of container managed components
    - Middle tier moved into ASP.NET
    - Using stored procedures instead of ad-hoc SQL
    - Uses server-side caching of ASP.NET
    - Many performance optimizations applied
- Most Java devotees find this highly unfair

TMC revisits the Pet Shop

- Re-implementation of J2EE version
  - 17 times performance increase
  - Second version showed some J2EE implementation equal .Net
  - Second version is a testimony to performance tuning



What does the comparison tell us?

- It is very difficult to make such comparisons
- That .Net has gained maturity extremely fast
- That the two frameworks are very similar
- The Devil is in the detail

So let’s look at some details

Comparing the stacks

Java vs. C#

- C# is an object oriented language of the C++/Java flavor
- Syntax similar to Java and C/C++.
- Quite an impressive design and care for details
- Java developers will feel comfortable
  - most of the time and frustrated when things are different
- MS says: “C# combines the power of VC++ with the ease of usage of VB”
  - Not really true:
    - C# is really powerful BUT
    - It is not easy to learn for non C++/Java programmers
- It is the language to learn if you are serious about .NET!

Java vs. C#

```
// This is a comment in Java code

class HelloWorld{
  public static void main(String[] args){
   for(int i= 1; i<= 100; i++)
      System.out.println("Hello!");
   }
}

// This is a comment in C#

using System;
class HelloWorld{
   static void Main(){
     for(int i=1; i<=100; i++)
        Console.WriteLine("Hello");
     }
   }
}
```

Primitives

- Java
  - int, float, double, etc.
  - Allocated on stack
  - Not an Object
  - Not extensible
- C#
  - int, float, double, etc.
  - structs
  - Allocated on stack
  - Inherited from object class
  - structs can implement interfaces
  - Cannot inherit from another class

```
// C#
struct Point
{
        int x;
        int y;
        Point(int x, int y)
        {
                this.x = x;
                this.y = y;
        }
}
```

Classes – Overriding Methods

- Java
  - All methods are implicitly virtual

```
class B
{
   public void foo() { }
}

class D extends B
{
   public void foo() { }
}

// D's foo() overrides B's foo()
```

- C#
  - Explicitly use 'virtual' and 'override' keywords

```
class B
{
   public virtual void foo() { }
}

class D:B
{
   public override void foo() { }
}

// D's foo() overrides B's foo()
```

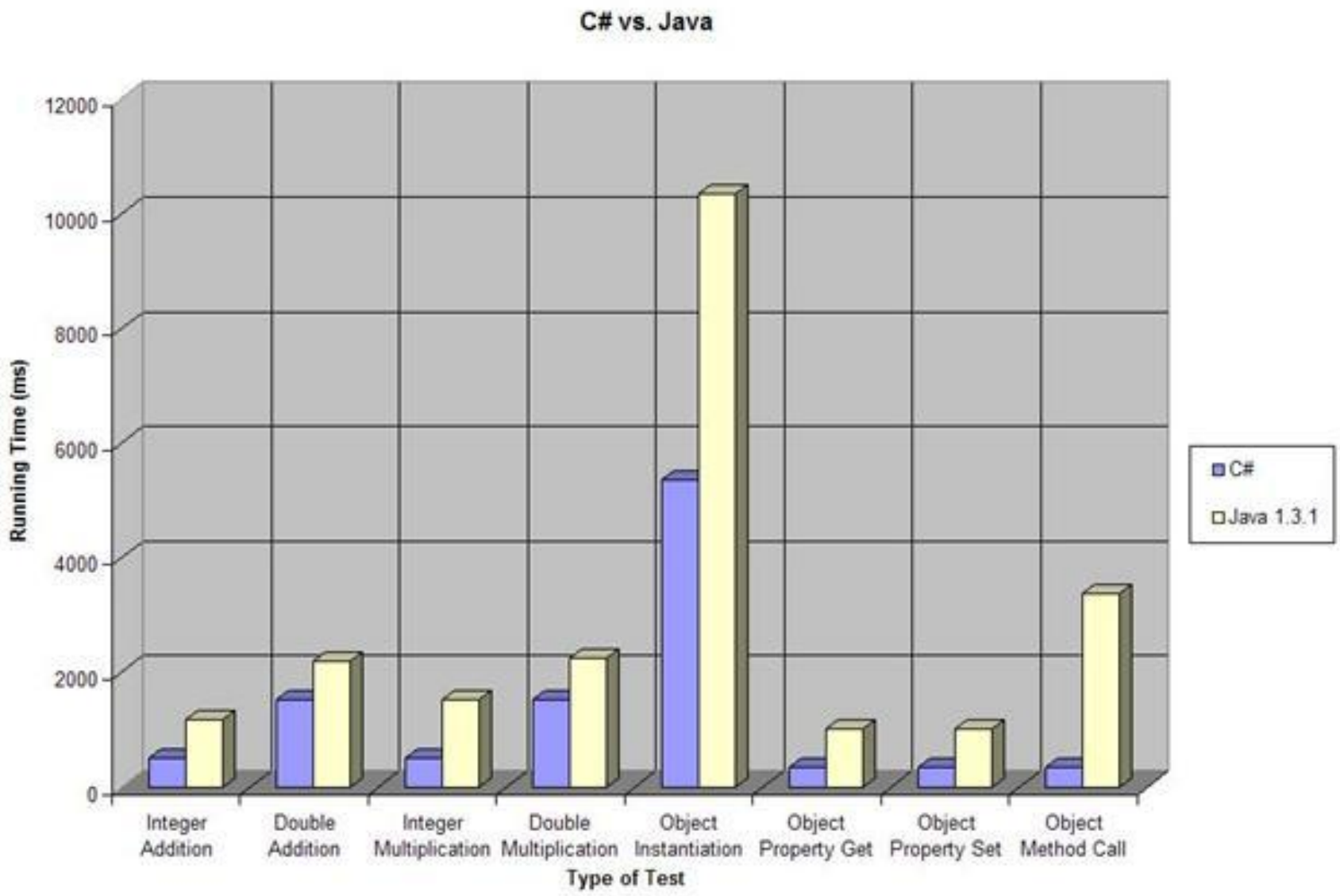
Exceptions

- Java
  - C++-style try/catch blocks
  - finally– action done even after an exception is caught
  - throws – methods identify what exceptions they throw
- C#
  - C++-style try/catch blocks
  - finally – same as Java
  - Does not support throws clause

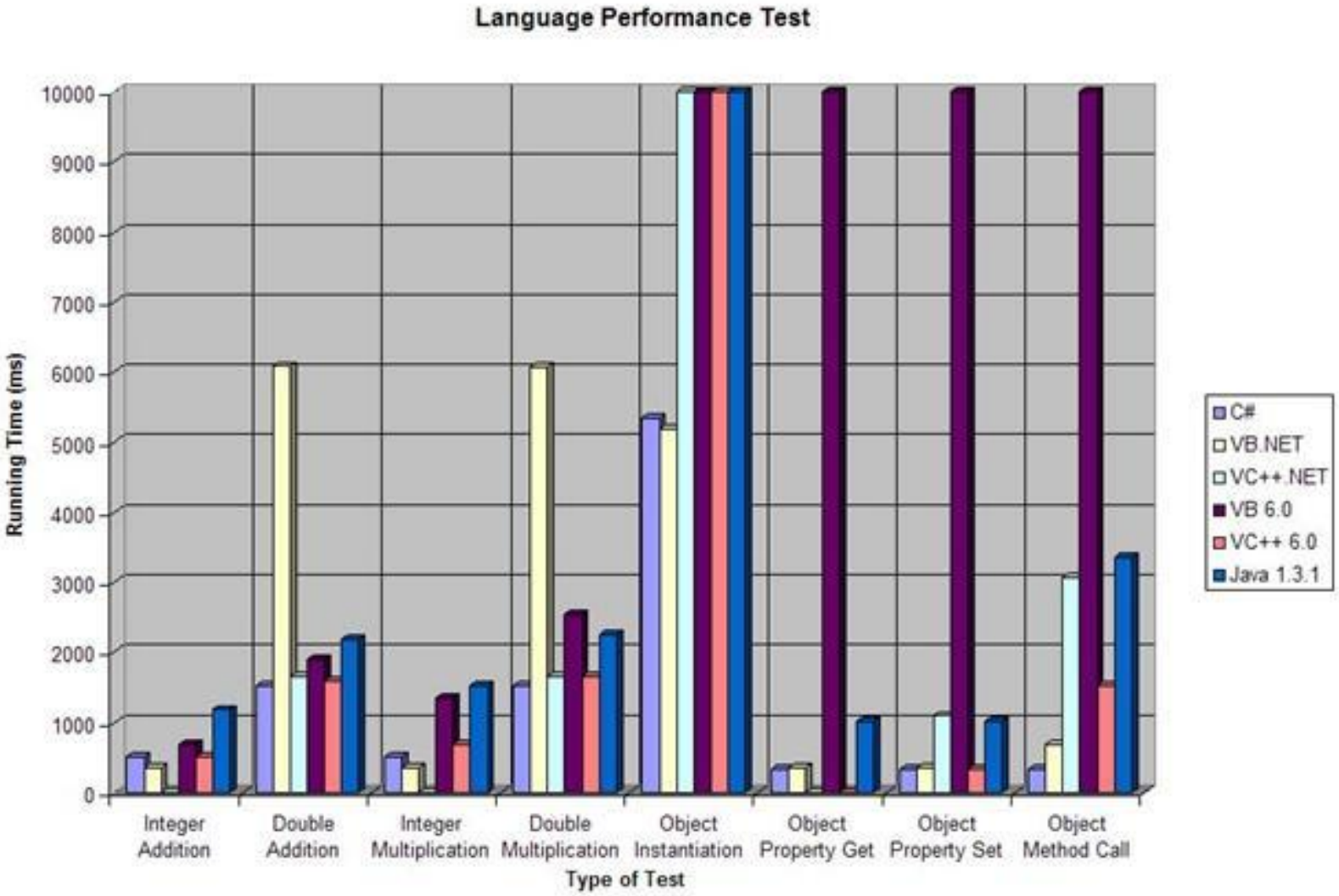
```
// Java – throws an IOException
public void myFunc(int a) throws IOException
{
        // Work…
}
```

```
// Java and C#
try {
        // Stuff…
}
catch {
        // Ack!
}
finally {
        // Always!
}
```

Java vs. C#


C# vs. Java
Running Time (ms)
12000
10000
8000
6000
4000
2000
0
Integer Addition
Double Addition
Integer Multiplication
Double Multiplication
Object Instantiation
Object Property Get
Object Property Set
Object Method Call
Type of Test
C#
Java 1.3.1

Java vs. C# vs. C++ vs. VB



Java vs. C# in High-performance computations

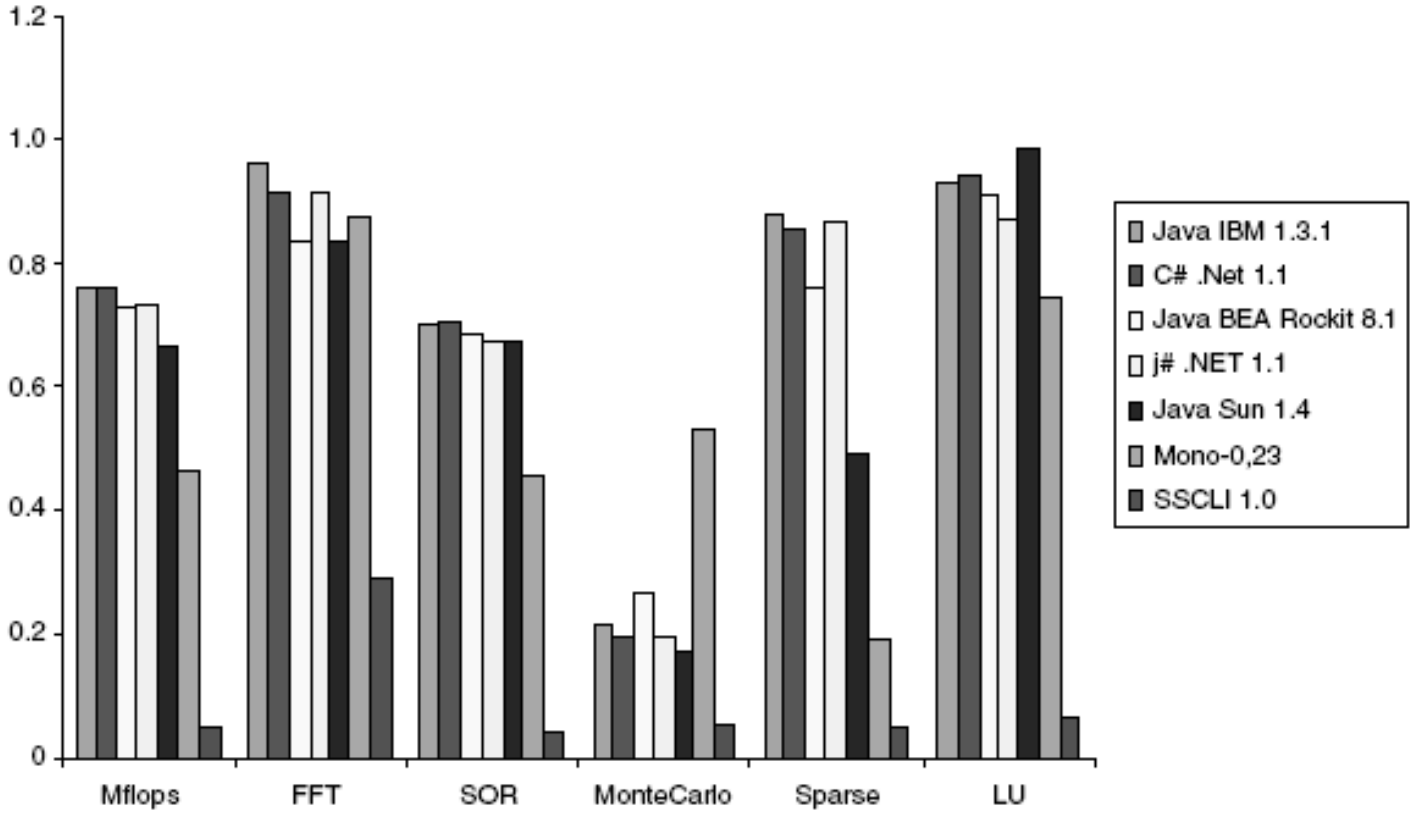
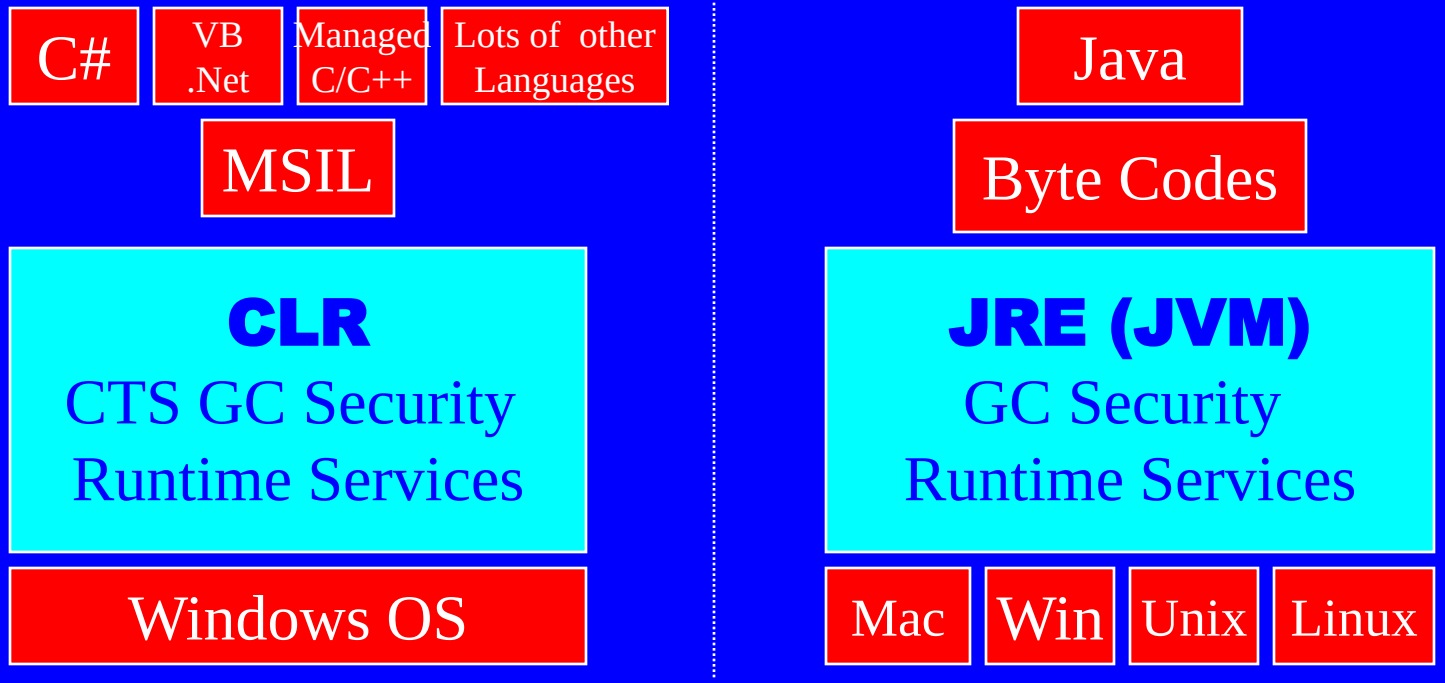


Fig. 11 *SciMark results for large memory model*

C# Features not in Java

- No automatic fall-through from one case block to the next
- Strongly-typed enums
- By reference calls are explicit at caller AND callee
- Method overrides are explicit
- Supports versioning
- Structs (value types)
- Integrated support for properties and events
- Can still use pointers with RAD language
- Can share data and use functionality with components written in many different languages

CLR vs JVM



Both are ‘middle layers’ between an intermediate language & the underlying OS

.Net on other platforms

- ECMA standardisation
  - ECMA 334 and 335
- The Mono Project
  - Shared Source Common Language Runtime
  - CLI, C# and Jscript
  - FreeBSD and Windows implementations
  - Linux port underway
- DOT GNU project
  - Portable .Net implementation
  - C# for both CIL and JVM

JVM vs. CLR

- JVM designed for platform independence
  - Single language: Java (?)
  - A separate JVM for each OS & device
- CLR designed for language independence
  - Multiple languages for development
    - C++, VB, C#, (J#)
    - APL, COBOL, Eiffel, Forth, Fortran, Haskel, SML, Mercury, Mondrian, Oberon, Pascal, Perl, Python, RPG, Scheme, SmallScript, …
    - Impressive usage of formal methods and programming language research during development
    - Impressive extensions for generics and support for functional languages underway
  - Underlying OS: Windows (?)

Java Byte Code and MSIL

- Java byte code (or JVML) is the low-level language of the JVM.
- MSIL (or CIL or IL) is the low-level language of the .NET Common Language Runtime (CLR).
- Superficially, the two languages look very similar.

JVML:

```
iload 1
iload 2
iadd
istore 3
```

MSIL:

```
ldloc.1
ldloc.2
add
stloc.3
```

- One difference is that MSIL is designed only for JIT compilation.
- The generic add instruction would require an interpreter to track the data type of the top of stack element, which would be prohibitively expensive.

JVM vs. CLR at a glance

	JVM	CLR
Managed execution environment	X	X
Garbage Collection	X	X
Metadata and Bytecode	X	X
Platform-abstraction class library	X	X
Runtime-level security	X	X
Runs across hardware platforms	X	?

J2EE 1.5

- J2EE (1.5) preview of 26.4.2004
  - Focus on ease of development
    - Generics and metadata as in J2SE 1.5 (more like C#)
    - Java Studio Creator tool (in beta from April 2004) (more like Visual Studio .Net)
  - Timeframe
    - To be discussed at JavaOne in June
    - Finalized in approximately one year
  - IBM push for truly open source Java
  - Others hesitate (even Open Source JBoss)

What about mobile?

- Web Services are fine as n-tier applications with UI provided through browser, but …
- On mobile devices WAP hasn't really caught on
- Even iMode hasn't caught on in Europe
- Renewed Thin/Thick client discussion
- Java applications on Mobile devices are reasonably successful
- Now Microsoft is moving fast into the field with .Net Compact Framework

Two Devices

Nokia 6600

Orange SPV E200

Two Devices


Menu
Phone
Services
Phoneb..
Calendar
Messag.
Settings
Pinboard
Log
Photos
Options
Exit
SIEMENS


FUJITSU SIEMENS
POCKET LOOX
Start
10:31
Friday, October 18, 2002
Owner: George Jiang
Eva's Birthday
Team Meeting
10:30AM-11:30AM (Conf room)
Lunch w/Sam & Griffin
12:00PM-1:00PM (Cafe)
Quarterly Business Review
1:00PM-3:00PM
ActiveSync: 18 Unread messages
1 Active task
New
Pocket PC

Building Mobile Solutions is harder

- Mobile device
  - Software infrastructure, hardware requirements
- Communication technology
  - On-/Offline scenario
  - Wireless Wide Area Networks/ Wireless Local Area Networks
  - Communication protocol
- Application architecture scenario
  - Thin/fat client
- Data management
  - Synchronisation
  - On-/offline capabilities
- Security issues
  - Dangers for mobile devices
  - Threats of communication technology
  - Danger of exposing enterprise data

Therefore ...

The Java vs. Net discussion goes mobile

- Java 2 Micro Edition (J2ME) is not ONE Java edition
- An J2ME compliant application consists of
  - Configuration
  - Profile (e.g. Personal, Mobile Information Device Profile (MIDP))
  - Application code
  - Vendor specific UI
- 3 Contenders to compare
  - Java 2 Micro Edition – Connected Device Configuration (CDC)
  - Java 2 Micro Edition – Connected Limited Device Configuration (CLDC)
  - Microsoft .NET Compact Framework

Student Project

- Group of Master Level Students (Hovedfag)
  - Bjørn D. Rasmussen
  - Casper S. Jensen
  - Jimmy Nielsen
  - Lasse Jensen
- Collaboration with DEIF
- Project Goals
  - Build end-to-end mobile client, wireless, webservices based application with back-end interface to SCADA
  - In Java (J2ME/J2EE) and in .Net
  - Compare the two solutions on a set of criteria

Basis of comparison

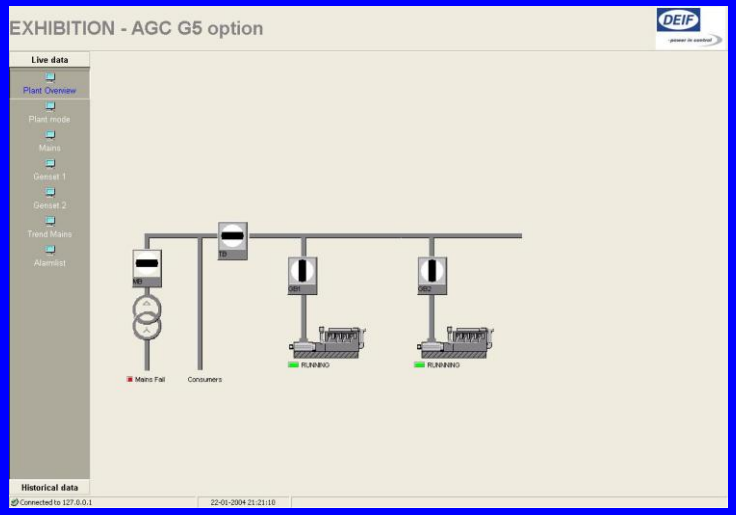
Objective measurements

- Lines of code
- Development tools
- Documentation
- Infrastructure
- Performance
- Architectural pattern
- Security
- Price
- Development time

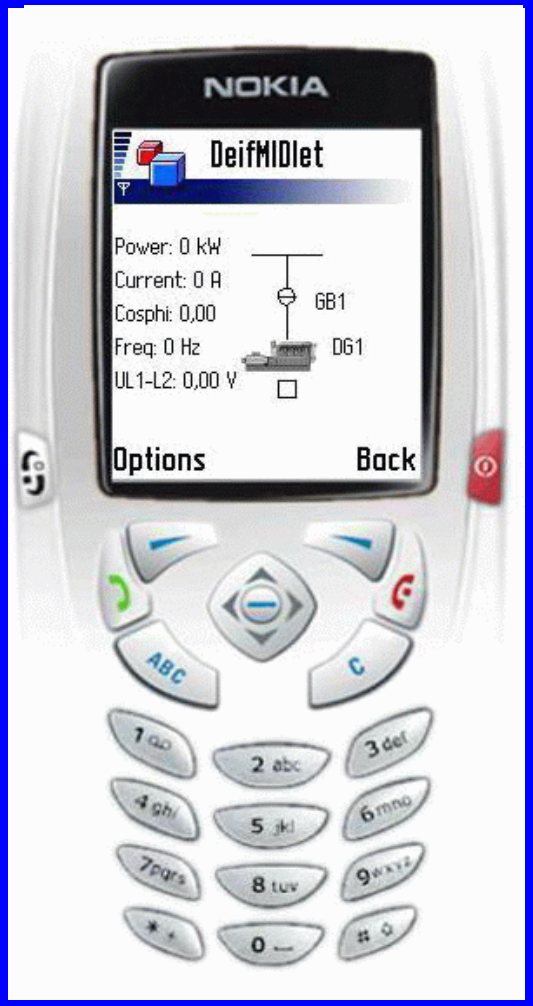
Subjective measurements

- Developer satisfaction
- End-user satisfaction

DEIF M-Vision
(SCADA up and running in 30 minutes)



SCADA on SmartPhones



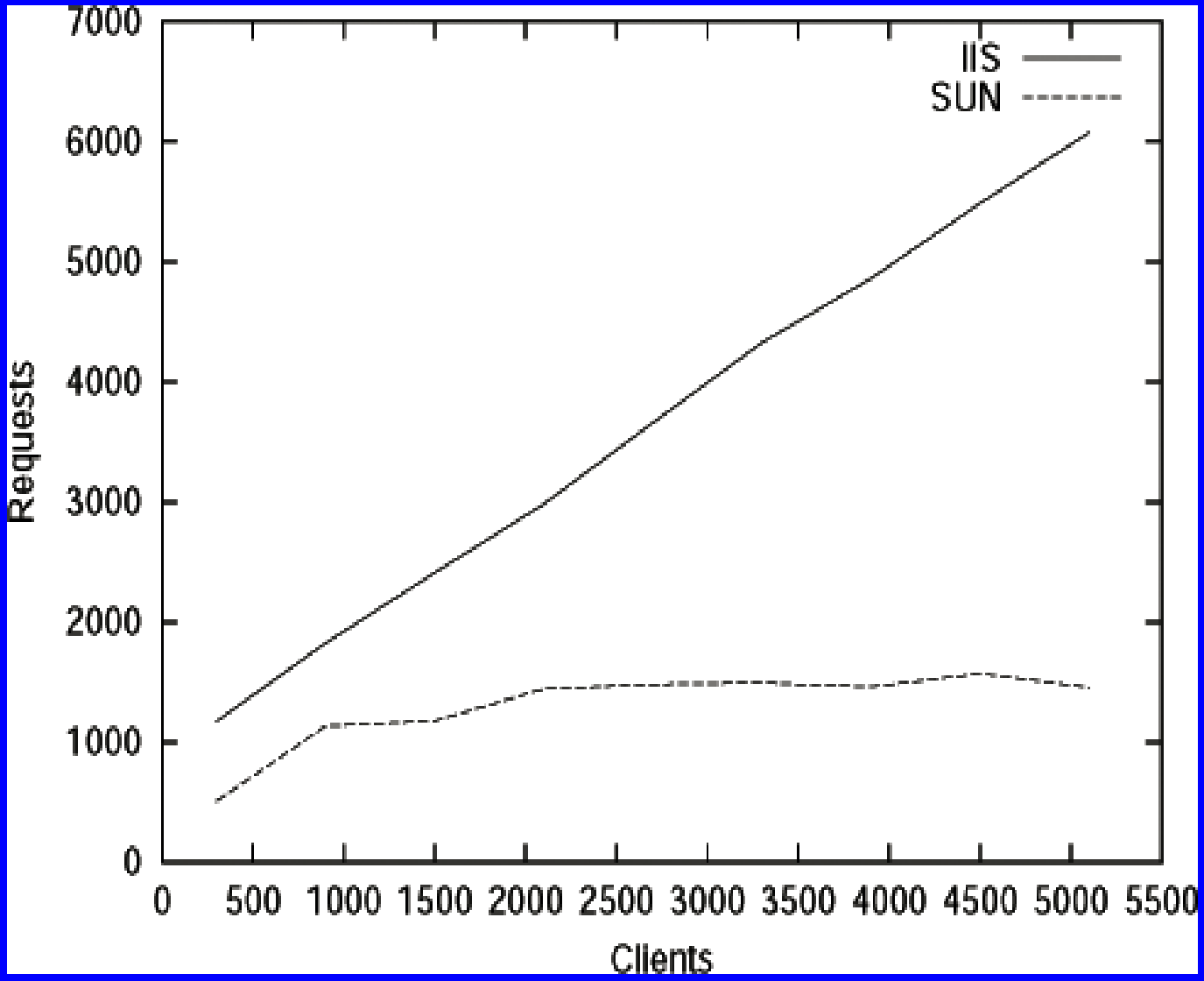
Development tools and Documentation

- Server-side is well supported by both Java and .NET IDEs
- On the client-side .NET IDEs benefit from the fact that .NET CF is so close to .NET (With Java there are separate IDEs for desktop and mobile application development)
- Compatibility problems between Java vendors
- Java IDEs are slow!
- C# is a richer/more complex language than Java
- Both Java and .NET have well documented API
- Web service documentation
  - .NET - MSDN
  - Java – Google
- Support for encryption of web services
  - .Net CF: HTTPS and SOAP extensions
  - J2ME: HTTPS, but only in CDC & MIDP 2.0

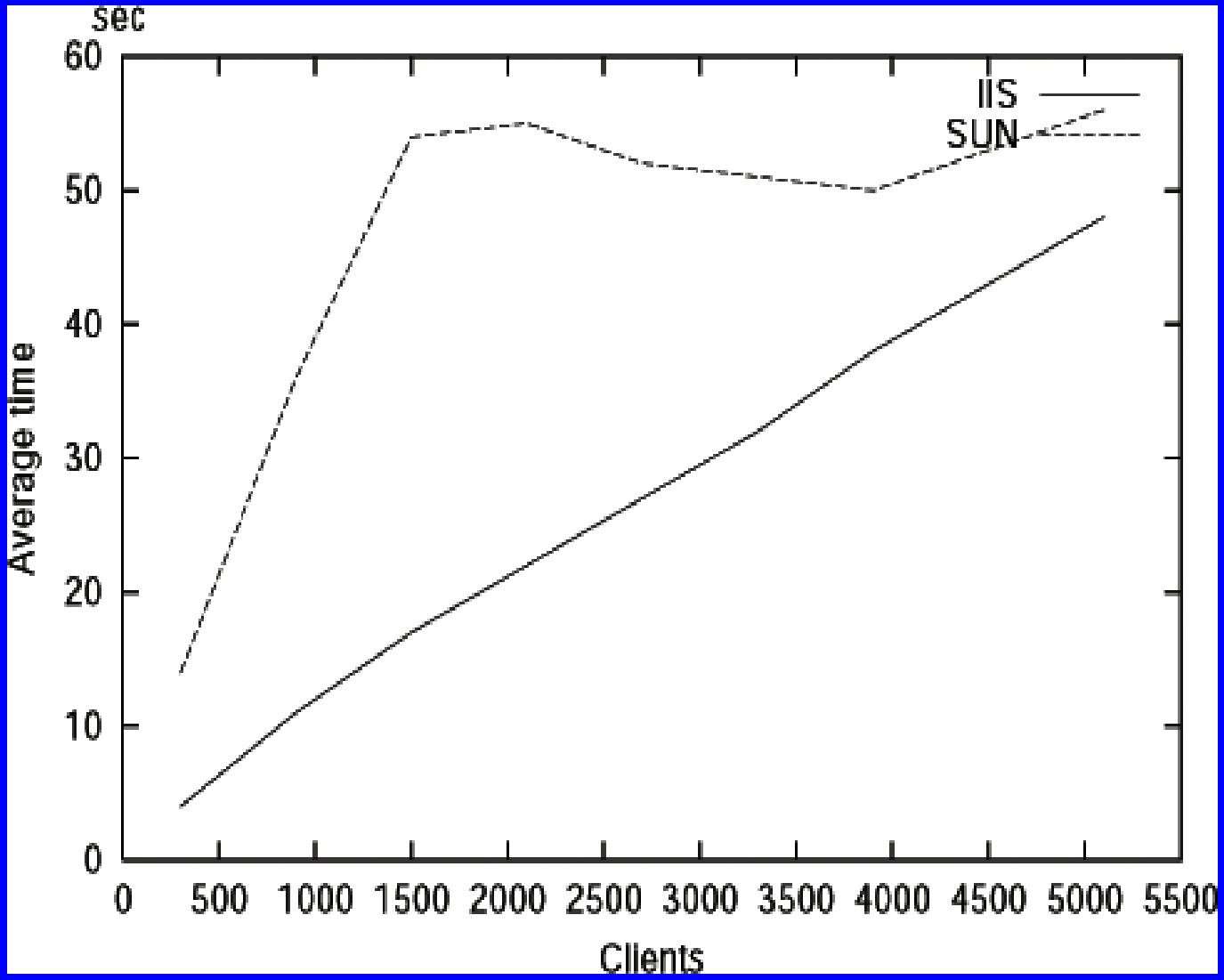
Performance

- Server-side web service performance study with:
  - Microsoft Application Server (v. 5.1)
  - Sun ONE Application Server (v. 7.0)
- Tested for:
  - Throughput
  - Failure rate
  - Average response time

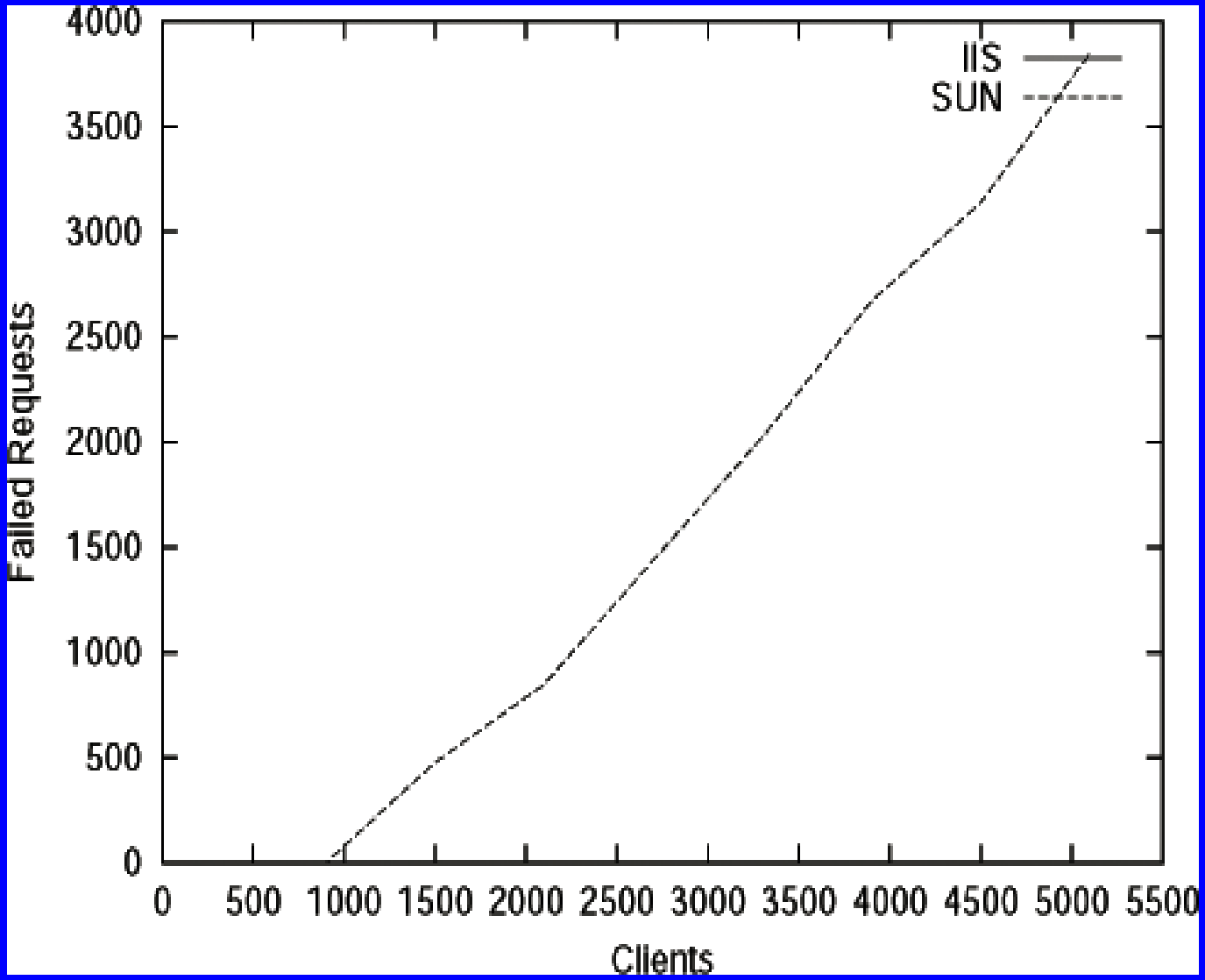
Performance - Throughput



Performance – Average response time



Performance – Failure rate



Development time

- Slow start-up when developing in Java
  - Jungle of web service implementations
  - IDE incompatibility problems
    - Emulators
    - kSOAP
- Trouble-free implementation in .NET

Subjective Measures

- Developer satisfaction
  - VS.NET integrates web service and mobile application development far better than any Java IDE
  - A subset of an API is a better solution for mobile devices instead of an entirely new API
- End-user satisfaction
  - DEIF would choose a .NET based solution since startup time is very important
  - DEIF only needs a limited number of IDE licenses
  - Extra price for a SmartPhone 2003 is insignificant compared to the value it gives
  - The SmartPhone 2003 is more ”fancy” than the Java phones

Students Conclusions

- We see .NET as a better designed framework because:
  - it eliminates the language barrier while also being platform independent
  - it makes only little distinction between desktop and mobile application development
- Sun's application server performance is very poor compared to IIS.
- License fees for a Java based solution are cheaper but .NET might catch up when considering development time

- We tried a combined .NET and Java solution but this has shown to be very troublesome!

My conclusions

- The two worlds are becoming more and more similar
  - But it seems that you have to work harder in Java than in .Net
  - .Net is currently driving technology transfer from Research to Product
- Windows generation at University now

- Watch-out in the mobile world
- Vodafone to offer Microsoft Smart phones
  - http://msmobiles.com/news.php/2504.html
  - Fed-up with Nokia promoting own brand, rather than operator brand

What do these comparisons tell us?

- It is very difficult to make such comparisons
- That .Net has gained maturity extremely fast
- That the two frameworks are very similar
- You will not be sacked for choosing the right J2EE application server ;-)
- The Devil is in the detail
  - C# is not Java
  - ADO.NET is not JDBC
  - CLR is not JVM
  - CF.Net smartphones are very different from Java Smartphones

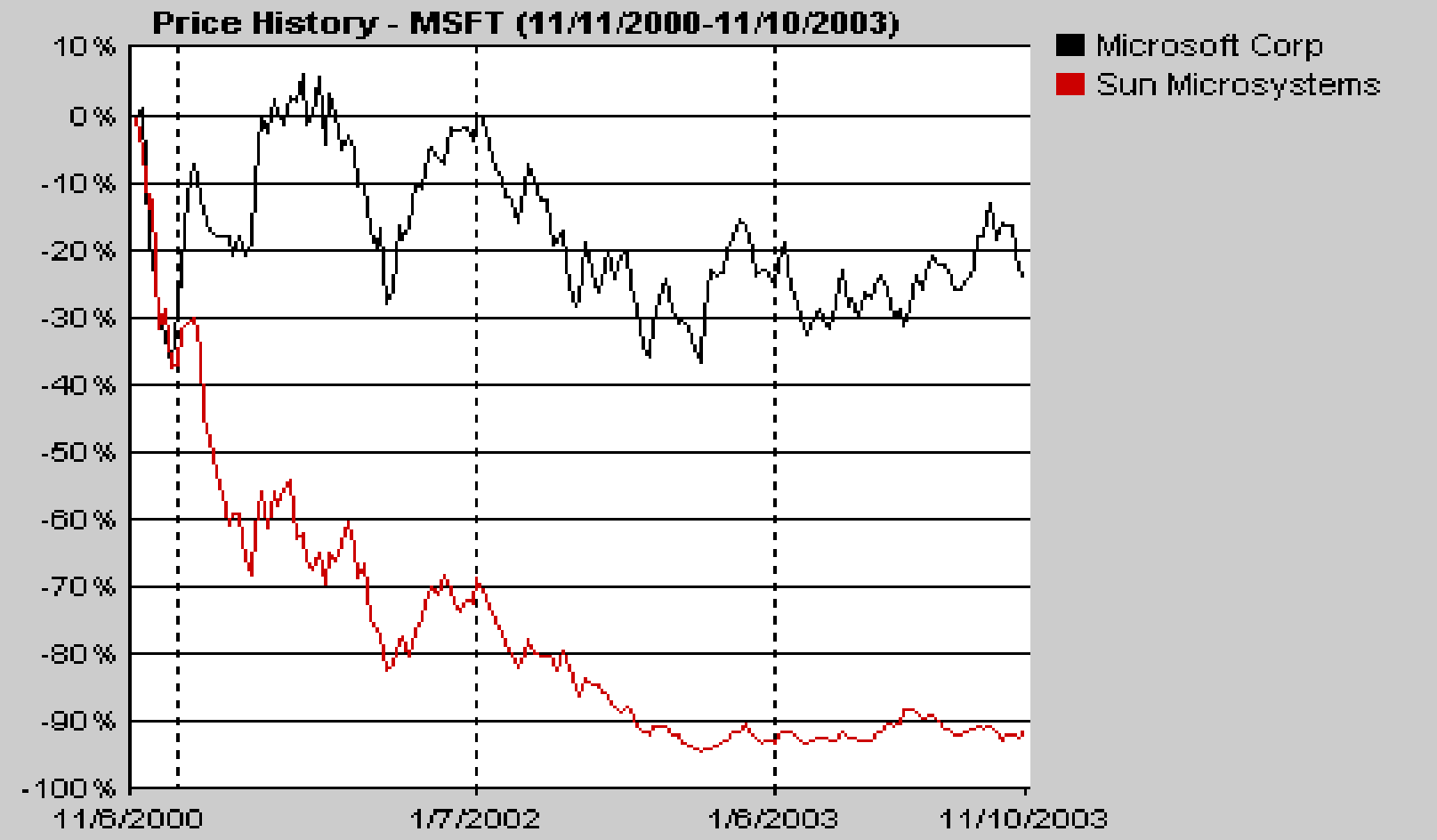
Choosing between Java and .Net

- The ultimate choice usually depends not on technical superiority, but on:
  - cultural/”religious”/political preferences
  - Skill set of your developers
  - Customer preference
  - Vendor relations

The future of Java and .Net

- The two frameworks are becoming more and more alike
- However:
  - .Net is Microsoft's core strategy
    - .Net will be part of OS in the next generation of Windows
    - Lot's of innovation in Longhorn – Avalon, indogo, WinFS
  - Is Java in Sun's core strategy?
    - Java 1.5 SE is very close to C# 2.0/.Net CLR
    - Sun Java Studio Creator somewhat close to VS.Net
    - Some talk of JVM as multi-language platform, but not really so far …
    - Sun in "Java as Open Source" battle with IBM

Sun vs. Microsoft Stock Prices



Choosing between Java and .Net

- You are most likely to be developing in both environments for the foreseeable future
  - Gartner Group: 30+% of all enterprise applications will have both J2EE and .Net code
  - Often IIS in front of J2EE infrastructure
  - Interoperability issues
    - Web Services (often too slow or doesn't support …)
    - J2EE/.Net bridge (IL -> JBC or JBC ->IL)
- Look out for "The third way"
  - Linux, Apache, MySQL, PhP, …
- Look out for disruptive technologies
  - It only takes one guy to get the right idea
  - and a small team to implement a completely new platform
  - and One large company to productise it
  - or a lot of grassroots …